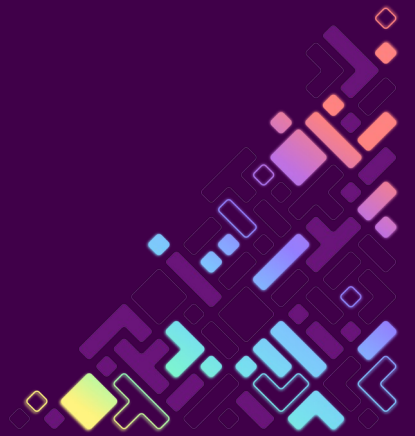
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

Using Yocto to debug embedded devices crashes

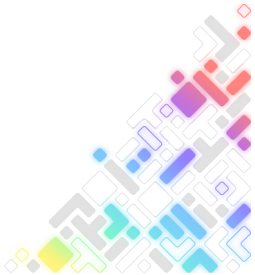


THE LINUX FOUNDATION

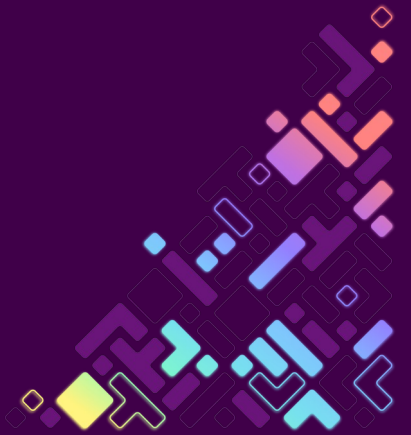
#ossummit @handle

Yocto optional features to debug crashes

- Debug symbol management
- Coredumps
- Kernel dumps



Debug symbol management



Debug symbol management

- Full debug symbol in image often not doable:

  - core-image-full-cmdline without dbg-pkgs: 70MB

```
poky/build$ ls -lh tmp/deploy/images/qemux86-64/
70M core-image-full-cmdline-qemux86-64.rootfs-20240907165026.tar.bz2
```

  - core-image-full-cmdline with EXTRA_IMAGE_FEATURES = “dbg-pkgs”: 319MB

```
poky/build$ ls -lh tmp/deploy/images/qemux86-64/
319M Sep  7 19:24 core-image-full-cmdline-qemux86-64.rootfs-20240907172408.tar.bz2
```

  - Even if possible slows down image creation, flashing, etc.

Debuginfod

- DISTRO_FEATURES “debuginfod” allows usage of debug symbol server
- “oe-debuginfod” runs e.g. on host or on dedicated server populated by build-server.
- “debuginfod” runs on target, and gdb fetches debug symbols using build-id from binary’s ELF section
- Build-id can be read using readelf -n:

```
$ readelf -n /usr/sbin/in.tftpd
...
Displaying notes found in: .note.gnu.build-id
  Owner                Data size          Description
  GNU                  0x00000014         NT_GNU_BUILD_ID (unique build ID bitstring)
    Build ID: 191f1fe2cf740fd811257541d7afa488289d1279
```

Minidebuginfo

- Created by Fedora in 2012 (Fedora 18). Supported by gdb and libdwarf from elfutils
- Contains only function names and addresses. Much smaller than full debug symbols

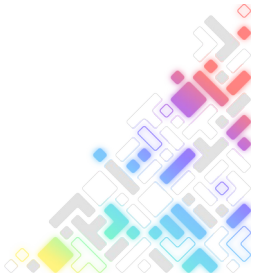
- Entire ELF file, compressed with LZMA, and injected as new ELF section “.gnu_debugdata” into binaries
- Yocto:
  - PACKAGE_MINIDEBUGINFO variable added in 2020 (available in Gatesgarth)
  - DISTRO_FEATURES minidebuginfo added in 2023 (Scarthgap)
    → enables minidebuginfo support in gdb/systemd: xz support in gdb and elfutils support in systemd

Minidebuginfo

- Core-image-full-cmdline:
  - No symbols: 70MB
  - Minidebuginfo: 73MB (4% overhead)

```
poky/build$ ls -lh tmp/deploy/images/qemux86-64/
73M core-image-full-cmdline-qemux86-64.rootfs-20240907180318.tar.bz2
```

  - Full debug symbols: 319MB (355% overhead)

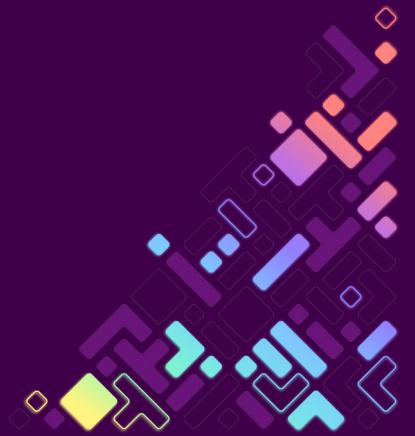


Companion debug filesystem

- IMAGE_GEN_DEBUGFS=”1” enables the debug filesystem (-dbg postfix)
- Can also be included with yocto SDK
- Useful when using gdbserver on target and pointing gdb on host to debug filesystem (set sysroot debugfs)

```
poky/build$ ls -lh tmp/deploy/images/qemux86-64/
306M core-image-full-cmdline-qemux86-64.rootfs-dbg-20240907191615-dbg.tar.bz2
```

Coredumps



Coredumps

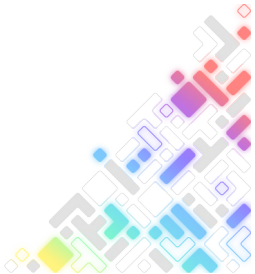
- When userspace software crashes, the kernel calls the coredump handler:

```
$ cat /proc/sys/kernel/core_pattern
|/usr/lib/systemd/systemd-coredump %P %u %g %s %t 9223372036854775808 %h
```

- Systemd’s coredump handler is systemd-coredump.
- Yocto: enabled e.g. with PACKAGECONFIG:append:pn-systemd = “coredump” in local.conf
- Disabled by default, but enabled by the minidebuginfo DISTRO_FEATURES

coredumpctl

- Systemd-coredump logs coredumps to the journal
- Full metadata available by filtering for coredump MESSAGE_ID:
  - journalctl -f MESSAGE_ID=fc2e22bc6ee647b6b90729ab34a250b1 -o verbose
- Coredumpctl retrieves information from the journal related to coredumps:
  - Coredumpctl list
  - Coredumpctl info
  - Coredumpctl gdb



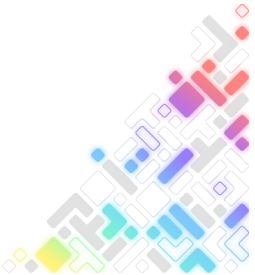
```
$ coredumpctl info 4335
           PID: 4335 (in.tftpd)
           UID: 0 (root)
           GID: 0 (root)
        Signal: 11 (SEGV)
[...]
       Message: Process 4335 (in.tftpd) of user 0 dumped core.
                Module in.tftpd from rpm tftp-5.2-41.fc39.x86_64
                Stack trace of thread 4335:
                #0  0x00007ffb2640ef8e __select (libc.so.6 + 0x112f8e)
                #1  0x00005578d0fde699 main (in.tftpd + 0x4699)
                #2  0x00007ffb2632414a __libc_start_call_main (libc.so.6 + 0x2814a)
                #3  0x00007ffb2632420b __libc_start_main@@GLIBC_2.34 (libc.so.6 + 0x2820b)
                #4  0x00005578d0fdfa15 _start (in.tftpd + 0x5a15)
                ELF object binary architecture: AMD x86-64
```

- Invoked twice. Once as kernel core handler (root), and the second time as systemd service systemd-coredump (socket activated, not privileged)

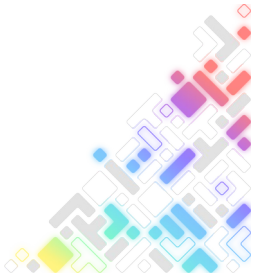
systemd/src/coredump/coredump.c:

```
r = sd_listen_fds(false);
/* If we got an fd passed, we are running in coredumpd mode. Otherwise we
 * are invoked from the kernel as coredump handler. */
if (r == 0) {
        return process_kernel(argc, argv);
} else if (r == 1)
        return process_socket(SD_LISTEN_FDS_START);
```

- sd_listen_fds() checks for file descriptors passed by the service manager
- See systemd-coredump@.service and systemd-coredump.socket for details



- Systemd-coredump tries to log a callstack to the journal
- Forks, drops capabilities, changes linux user and mount namespaces, and then calls elfutils libdwarf to parse symbols
- Libdwarf can parse full debug symbols and minidebuginfo symbols



Caveat 1

- Since v250, the fork attempts to change user namespace and fails silently on kernels without user namespace support (CONFIG_USER_NS)
- I created a PR to systemd that needs more work to get merged.
- The default linux-yocto has this config enabled, but not e.g. with KERNEL_FEATURES+= “small”

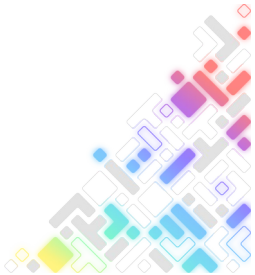
```
r = safe_fork_full("(sd-parse-elf)",
                   (int[]){ fd, error_pipe[1], return_pipe[1], json_pipe[1] },
                   FORK_RESET_SIGNALS|FORK_CLOSE_ALL_FDS|
FORK_NEW_MOUNTNS|FORK_MOUNTNS_SLAVE|
FORK_NEW_USERNS|FORK_WAIT|FORK_REOPEN_LOG,
                                 NULL);
```

Caveat 2

- For security reasons, systemd-coredump service file contains:
  - “ProtectHome=true”
  - “PrivateTmp=yes”
- A patch was merged to systemd and poky last week to change it to ProtectHome=read-only.
- So it now works under /home (still no symbolicated call stacks under /root and /tmp)

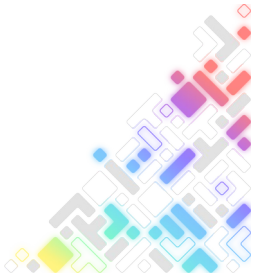
Caveat 3

- When a systemd service crashes, the kernel calls the coredump handler etc.
- It can easily take 10 seconds to write the coredump to the filesystem for processes consuming a lot of memory
- When reaching systemd's TimeoutStopSec, SIGKILL is sent to the process, which terminates it immediately, and the coredump is truncated
- Possible fixes:
  - Increase TimeoutStopSec (not always possible to delay restart in production)
  - Storage=none in coredump.conf (the file is not stored, but the journal contains the logs)
  - Reduce which memory pages are being dumped: kernel parameter coredump_filter
  - Madvise() with parameter MADV_DONTDUMP

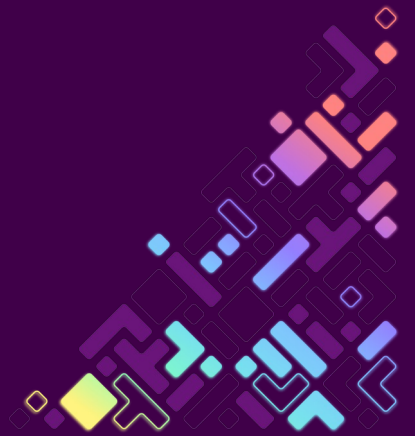


Coredumps

- Quite powerful combined with a crash-reporter like e.g. GlitchTip. Used to collect and aggregate the data of coredumps per function name, binary name, etc.
- Coredumps + minidebuginfo + crash-reporter are a simple solution not needing to manage full debug symbols automatically.
- You can also store debug-symbols and create symbolicated call-stacks using the crash-reporter Back-End, in case simple symbolicated minidebuginfo call-stacks are not suitable
- Meta-openembedded also provides support for breakpad + minidumps, in case full coredumps are not adapted to your project constraints (storage / bandwidth)
- User privacy concerns for production need to be taken into account. Best suited for testing stage.

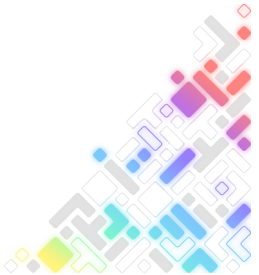


Using Kernel Dumps with Yocto



Kernel dumps

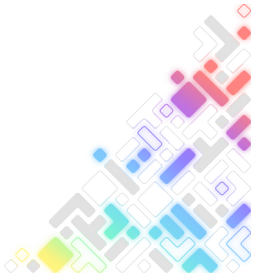
- Feature of the Linux kernel that creates crash dumps in the event of a kernel crash / panic
- Boots second kernel using kexec, which creates file /proc/vmcore
- Vmcore: ELF core file which contains all the physical memory used by the kernel
- Contains ELF notes about processes running on each CPU
- Also contains dmesg logs



Kernel dumps

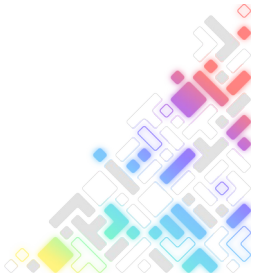
- The second kernel (capture kernel) uses its own memory, in order to preserve the crashed kernel memory

- Kernel command-line: “crashkernel=128M@256M”



Kernel dumps

- Very useful to solve kernel crashes which are hard to reproduce, during testing phases of a product.
- However, due to extra memory requirement and extra reboot time, often not appropriate for production where you typically rather want “panic=-1”



Kernel dumps

- Capture kernel needs to be loaded before a panic occurs.
- Loaded with “kexec -p” manually:

```
# cat /sys/kernel/kexec_crash_loaded
0
# kexec -p /boot/vmlinux-6.6.45-yocto-standard --append='maxcpus=1
reset_devices irqpoll swiotlb=0 root=/dev/vda rw systemd.unit=multi-user.target'
# cat /sys/kernel/kexec_crash_loaded
1
```

Kernel dumps

- Kexec can fail in various ways:
  - Issue with crashkernel command-line parameter:

```
# cat /var/log/messages | grep crashkernel
Sep  1 19:25:38 qemux86-64 kernel: crashkernel reservation failed - memory is
in use.
```

  - Missing kernel KCONFIG_KEXEC option:

```
kexec_load failed: Function not implemented
```

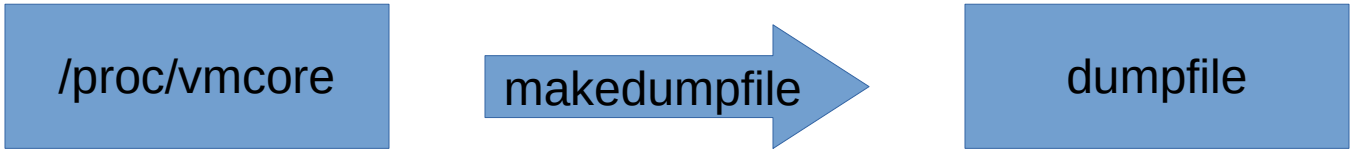
How to enable kdump in yocto

- Poky provides systemd's kdump.service, which runs /usr/libexec/kdump-helper:

```
# poky/meta/recipes-kernel/kexec/kexec-tools/kdump
start)
        if [ -s /proc/vmcore ]; then
                do_save_vmcore
                reboot
        else
                # executes kexec to pre-load the capture kernel:
                do_start
```

Kernel dumps

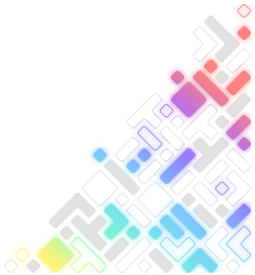
- Kdump-helper also calls makedumpfile when the kernel panics



- Per default, creates file in kdump-compressed format.
- “crash” can read this format

Kernel dumps

- Makedumpfile can also generate files in elf format (if only gdb can be used for some reason), but the files are a lot bigger
- Makedumpfile removes unnecessary pages (with zeroes, free pages, userspace data, etc)
- Makedumpfile.conf to filter certain modules e.g. for security concerns or to reduce size of dumps
- Makedumpfile uses vmcoreinfo from vmcore file (metadata like page size, kernel randomization offset, etc.)

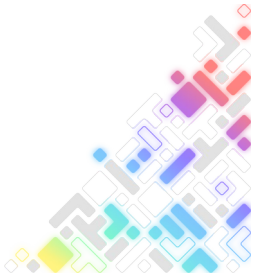


Kernel dumps: enabling it in yocto

- Kexec-tools from poky. Crash and makedumpfile from meta-openembedded:

```
# kexec-tools to load the kdump kernel
# crash to analyze the kernel dumps
# makedumpfile to create the filtered and compressed kernel dumps
# gdb as alternative to crash (but works only on raw kernel dumps in ELF format)
IMAGE_INSTALL:append = " kexec-tools crash makedumpfile gdb"
```

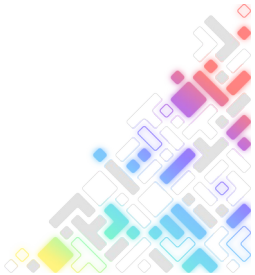
- Kexec-tools needs makedumpfile, but no RDEPENDS because it is in meta-openembedded



Kernel dumps: enabling it in yocto

- Kernel configuration changes (CONFIG_DEBUG_INFO is enabled per default in poky):

```
$ cat kdump-6.6.cfg
CONFIG_KEXEC_CORE=y
CONFIG_KEXEC=y
CONFIG_CRASH_DUMP=y
CONFIG_CRASH_HOTPLUG=y
CONFIG_CRASH_MAX_MEMORY_RANGES=8192
CONFIG_EFI_RUNTIME_MAP=y
CONFIG_PROC_VMCORE=y
# CONFIG_PROC_VMCORE_DEVICE_DUMP is not set
```

Kernel dumps: enabling it in yocto

- Need to use KERNEL_IMAGETYPE=”vmlinux” (no bzImage etc.)
- Debug-symbols accessible by makedumpfile, e.g. in linux-yocto:

```
INHIBIT_PACKAGE_STRIP=”1”
INHIBIT_PACKAGE_DEBUG_SPLIT=”1”
```

- Increase QB_MEM sufficiently when testing with qemu (e.g. “-m 2048”)
- IMAGE_ROOTFS_EXTRA_SPACE:append = " + 4000000"

Kernel dumps: enabling it in yocto

- Add crashkernel command-line parameter:

```
QB_KERNEL_CMDLINE_APPEND:append = " crashkernel=512M@256M"
```

Kernel dumps: enabling it in yocto

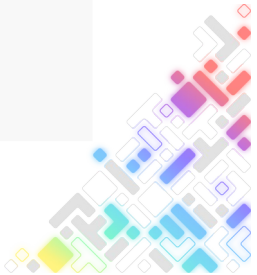
- Not an official yocto DISTRO_FEATURE, so no testing during Yocto releases
- For some unknown reason kexec was failing with kernel 6.10 during my testing (at least with ppc64 the kdump kernel crashes after the reboot)

```
PREFERRED_VERSION_linux-yocto = "6.6%"
```

- Kexec also failing with qemux86-64 “Invalid memory segment 0x1000000 – 0x272bfff”. Didn’t investigate further and tested with qemuppc64 instead.

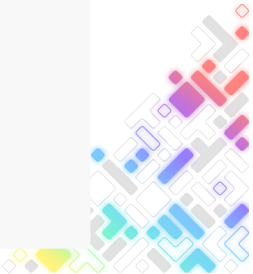
Kernel dumps: testing in qemu

```
root@qemuppc64:~# echo c>/proc/sysrq-trigger
[  108.917746] sysrq: Trigger a crash
[  108.920204] Kernel panic - not syncing: sysrq triggered crash
[  108.922040] CPU: 0 PID: 224 Comm: sh Kdump: loaded Not tainted
6.6.45-yocto-standard #1
...
[  108.928095] --- interrupt: 3000
I'm in purgatory
[    0.000000] radix-mmu: Page sizes from device-tree:
```

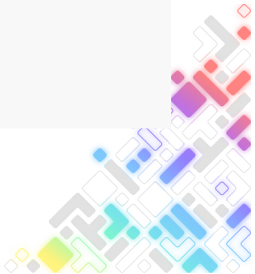



Kernel dumps: testing in qemu

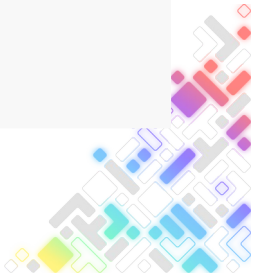
```
root@qemuppc64:~# cat /proc/cmdline
maxcpus=1 reset_devices irqpoll swiotlb=0 root=/dev/vda rw
systemd.unit=multi-user.target elfcorehdr=304064K

root@qemuppc64:~# dmesg | grep -i kdump
[    1.558380] 2048 entries for the kdump boot

root@qemuppc64:~# ls -lh /proc/vmcore
-r-------- 1 root root 769M Sep  2 14:14 /proc/vmcore
```



```
root@qemuppc64:~# makedumpfile -d 31 -D -x /boot/.debug/vmlinux-6.6.45-yocto-standard /proc/vmcore /var/crash/2024-09-02/test
...
...
The dumpfile is saved to /var/crash/2024-09-02/test.

makedumpfile Completed.
```

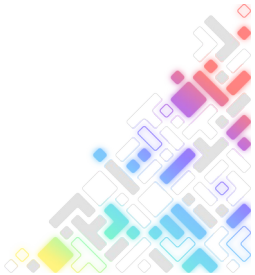



```
# Default behavior of makedumpfile: no filtering

root@qemuppc64:~# ls -lh /proc/vmcore
-r-------- 1 root root 1.6G Sep  2 18:59 /proc/vmcore
root@qemuppc64:~# ls -lh /var/crash/2024-09-02/test
-rw------- 1 root root 1.6G Sep  2 18:49 /var/crash/2024-09-02/test

# With filtering -d 31, only 16MB left in this test:

root@qemuppc64:~# ls -lh /var/crash/2024-09-02/dump-filtered-31
-rw------- 1 root root 16M Sep  2 19:04 /var/crash/2024-09-02/dump-filtered-31
```

Kernel dumps: testing in qemu

```
root@qemuppc64:~# crash /boot/vmlinux /var/crash/2024-09-02/dump-filtered-31

crash 8.0.4
...
```

```
crash> bt
[...]
 DSISR: 0000000000000000     Syscall Result: 0000000000000000
 [NIP  : __crash_kexec+264]
 [LR   : panic+420]
 #0 [c000000003ee7930] __crash_kexec at c000000000211af8
 #1 [c000000003ee7af0] panic at c000000000105018
 #2 [c000000003ee7b90] sysrq_handle_crash at c000000000ad8650
 #3 [c000000003ee7bf0] __handle_sysrq at c000000000ad8fe0
 #4 [c000000003ee7c90] write_sysrq_trigger at c000000000ad99dc
 #5 [c000000003ee7cc0] proc_reg_write at c0000000005320ac
 #6 [c000000003ee7cf0] vfs_write at c0000000004774c4
 #7 [c000000003ee7dc0] ksys_write at c000000000479650
 #8 [c000000003ee7e10] system_call_exception at c00000000002d8e8
 #9 [c000000003ee7e50] system_call_vectored_common at c00000000000cbdc
crash>
```

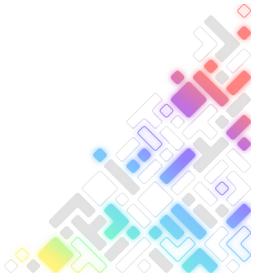
Kernel dumps: testing in qemu

- Very versatile tool: ps, mount, dev, etc. Can even generate ftrace files with a plugin.

```
crash> ps
     PID    PPID  CPU       TASK        ST  %MEM     VSZ      RSS  COMM
>      0       0   0  c00000000272f400  RU   0.0       0        0  [swapper/0]
       0       0   1  c000000003574400  RU   0.0       0        0  [swapper/1]
       1       0   1  c000000003520000  IN   0.8   23808    16256  systemd
       2       0   1  c000000003521100  IN   0.0       0        0  [kthreadd]
       3       2   0  c000000003522200  IN   0.0       0        0  [pool_workqueue_]
       4       2   0  c000000003523300  ID   0.0       0        0  [kworker/R-rcu_g]
       5       2   0  c000000003524400  ID   0.0       0        0  [kworker/R-rcu_p]
```

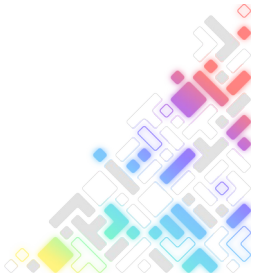
Kernel dumps: caveat

- “crash” supports KASLR only for x86-64 at the moment
- Thus, best to add “nokaslr” for internal debug images when the kernel is compiled with KASLR (CONFIG_RANDOMIZE_BASE)
- “nokaslr” not suitable for production because of security concerns



Kernel dumps

- Similar to userspace coredumps, ideally used with a crash-reporter early in a project, kernel dumps make it significantly easier to solve kernel crashes for Yocto projects



THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

EUROPE